Evolutionary Design and Simulation of a Tube Crawling Inspection Robot

Michael Craft and Dr. Tom Howsman
Dynamic Concepts, Inc.
Madison, AL 35758
e-mail: mcraft@dynamic-concepts.com

Dan O'Neil
NASA, Marshall Space Flight Center
Huntsville, AL 35812
e-mail: dan.oneil@msfc.nasa.gov

Keywords
Evolutionary robotics, robot simulation, genetic algorithm

Abstract

The Space Robotics Assembly Team Simulation (SpaceRATS) is an expansive concept that will hopefully lead to a space flight demonstration of a robotic team cooperatively assembling a system from its constitutive parts. A primary objective of the SpaceRATS project is to develop a generalized evolutionary design approach for multiple classes of robots. The portion of the overall SpaceRats program associated with the evolutionary design and simulation of an inspection robot's morphology is the subject of this paper. The vast majority of this effort has concentrated on the use and modification of Darwin2K, a robotic design and simulation software package, to analyze the design of a tube crawling robot. This robot is designed for carrying out inspection duties in relatively inaccessible locations within a liquid rocket engine similar to the SSME. A preliminary design of the tube crawler robot was completed, and the mechanical dynamics of the system were simulated. An evolutionary approach to optimizing a few parameters of the system was utilized, resulting in a more optimum design.

1.0 Introduction

Efforts to develop a generalized approach using evolutionary methodologies for the design of specific types of robots have been jointly pursued by Marshall Space Flight Center and Dynamic Concepts, Inc. The specific efforts documented in this paper concentrate on using a genetic algorithm in concert with dynamic simulation to optimize the structural design, or morphology, of a robot. The mission of the robot was to crawl through a small tube independent of the direction of gravity. Potentially, the robot was designed to carry out inspection duties in relatively inaccessible locations within a rocket engine similar to the Space Shuttle Main Engine (SSME). The vast majority of the efforts documented herein have been applied to the use and modification of Darwin2K, a robotic design software toolkit, to analyze the design of the robot. A preliminary design of the tube crawler robot has been completed, and the mechanical dynamics of the system have been simulated. An evolutionary approach to optimizing a few parameters of the design has been utilized, resulting in a more optimum design. Details of the Darwin2K software and the tube crawling robot design are presented in this paper.

2.0 Software for the Evolutionary Design of Robotic Hardware

An investigation into the availability of software for the design of robotic morphologies was conducted early in the contract period of performance. While there are several research centers active in the application of evolutionary strategies to the design of robot morphology [1] [2], there is very little software available to the public at this time. The available software consists primarily of general purpose evolutionary algorithms, and typically lacks the important specializations and underlying physics models required for robotic morphology design.

A notable exception to the general purpose software is the Darwin2K software package [3]. Originally authored by Dr. Chris Leger as part of his Ph.D. program at Carnegie Mellon University, development of Darwin2K is now being performed by a small number of

researchers and is hosted on the open source development site SourceForge.net. DCI has decided to utilize the Darwin2K software and has installed the software on several of its workstation Linux PC's. The installation of the software is not a trivial matter, and requires some familiarity with Unix, OpenGL, Xforms, and Perl.

3.0 Darwin2K Overview

Darwin2K is an open source software suite of tools for robot simulation and design optimization. Darwin2K allows a user to define a robot using a simple tree structure, draw from (or add to) a library of pre-defined robotic joints, define the robot's terrain and performance metrics, and simulate the kinematics and dynamics of the robot. An evolutionary design algorithm (genetic algorithm) is implemented to allow the user to optimize the robot kinematics, dynamics, components, and controller parameters. Darwin2K is a series of object-oriented applications created in C++ and is intended to be compiled and executed on Unix-based operating systems, such as Irix or Linux. It should be noted that Darwin2K is not an "end user" software product. In order to implement any type of complex robotic design, the user is required to either modify or add to the Darwin2K source code. Further information about Darwin2K may be found at www.darwin2k.com.

4.0 Design and Simulation using Darwin2K

A significant effort has been made towards using the Darwin2K software suite to develop the design of a robot capable of crawling through a tube independent of the direction of gravity. In order to quickly develop a prototype robot, an existing robot design was modified to allow for the robot's legs to simultaneously make contact with an upper and lower surface. A diagram of this robot configuration is shown in Figure 1. While this design appears to meet the mission requirements, it results in a complex configuration with 24 separate joints. In order for this robot to be effective, a controller would

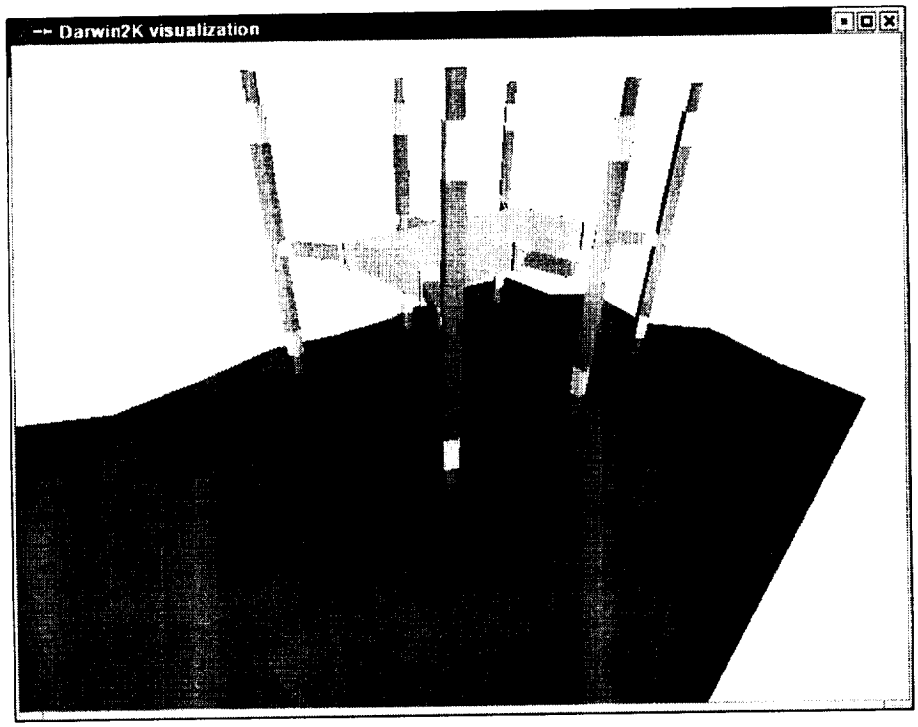

Figure 1: Modified Darwin2K Hexbot on Random Terrain

have to be developed to coordinate the complex motion of the robot's joints.

A different, less complicated design of a robot to perform the mission has been pursued. This robot design, called "UmbrellaBot" throughout the remainder of this document, consists of only three controllable joints. Two of the joints are prismatic beams that control a mechanism that extends and retracts a series of legs. The third joint is another prismatic beam connected between the other two joints in sequence, thus allowing the robot's body to expand and contract like an inchworm. A diagram presenting the preliminary design of the robot, referenced as "UmbrellaBot" in this report, is shown in Figure 2.

Unfortunately, the leg design shown in Figure 2 results in a series of "closed chain" mechanisms, or mechanisms in which the dynamics calculations loop back upon themselves. In its unmodified form, Darwin2K cannot simulate the dynamics of such mechanisms. The changes made to Darwin2K to implement the UmbrellaBot configuration are discussed in the following sections.

5.0 Darwin2K Modifications

In Darwin2K, robots configurations are "assembled" using a map to connect the series of robotic joints (e.g., a prismatic beam), links (e.g., a hollow tube), bases, and end effectors to form a single design. Obviously, Darwin2K implements a finite number of types of components stored in a library. In order to build certain specific configurations, it is necessary to add required components, controllers, mechanisms, or other classes to the Darwin2K source code. Additional mechanical links and joints were added to the Darwin2K component library in order to construct the UmbrellaBot. Perhaps the most difficult aspect of this project was the implementation of the closed-chain mechanisms discussed previously. A customized evaluator class was derived to implement the various constraints that result in the umbrella leg mechanisms. A diagram depicting the evaluator constraints implemented to form the leg mechanisms is presented in Figure 3. Finally, a controller was developed to coordinate the movement of the UmbrellaBot.

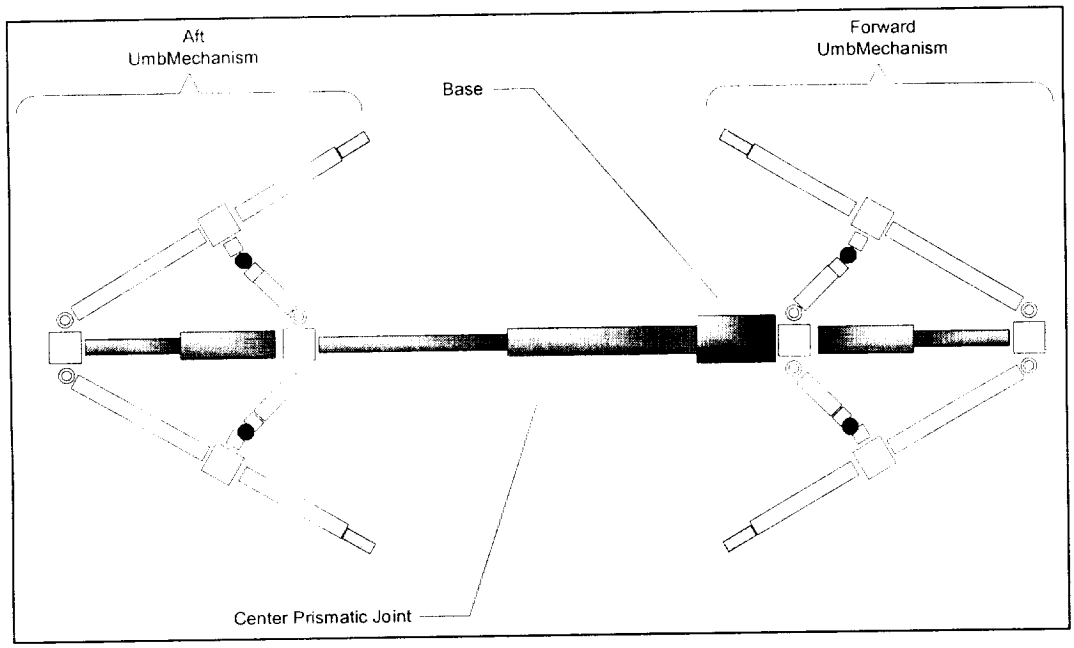

Figure 2: UmbrellaBot Conceptual Design

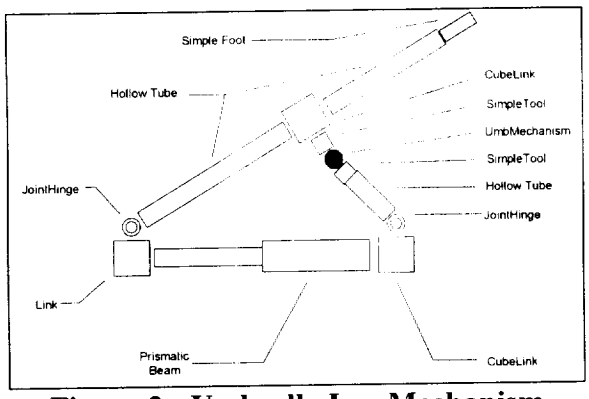

Figure 3: Umbrella Leg Mechanism

6.0 Simulation and Evaluation Procedure

In order to evaluate a robot configuration using Darwin2K, a specific sequence of actions must be followed in order to ensure that the program will properly analyze the correct configuration. This sequence of actions is discussed in the following sections, along with examples of the input required for each step and examples of graphics generated by the Darwin2K suite of tools.

6.1 Configuration Display

When building a prototype in Darwin2K, it is often useful to check the robot configuration to ensure the joints and links are being constructed as intended. The program "displayCfgGL" is designed to accomplish this task. The program displayCfgGL graphically renders the robot configuration and allows the user to view the specified configuration from virtually any angle, check the motion of the robotic joints, and inspect the dynamic chains formed by Darwin2K. The displayCfgGL program is activated by typing "`displayCfgGL config.l`" at the prompt of a Unix console window, where the file "config.l" is the configuration file of the desired robot.

6.2 Simulation

Before a robot configuration can be analyzed and refined by the Darwin2K population manager, it is necessary to simulate the dynamics of the robot's baseline configuration to ensure that the controller functions properly, that the robot does not collide with any objects, and that the robot can achieve its goals. There are two dynamic simulation programs included in the most recent version of Darwin2K: the "d2kSimGL" simulation and the "evStandaloneGL" simulation. Both programs require a configuration file (config.l), an evaluation parameter file (eval.p), and a terrain definition file (terrain.dat). However, the evStandaloneGL program also requires a population manager parameter file (pm.p), which defines the robot's performance metrics. The d2kSimGL program is activated by typing "`d2kSimGL eval.p config.l`" at the prompt of a Unix console window, while evStandaloneGL is activated by typing "`evStandaloneGL pm.p eval.p config.l`" at the prompt of a Unix console window. Both programs graphically render the simulated dynamics of the robot. A diagram presenting an example of the graphics generated by the Darwin2K dynamic simulations is shown in Figure 4.

6.3 Population Manager

The Darwin2K population manager program, "pmStandalone," is used to generate many different robot configurations based on the baseline configuration, evaluate each configuration based on the performance metrics defined in the population manager parameter file (pm.p), and determine the optimal robot configuration for each performance metric. The evolutionary analysis features of Darwin2K are contained in the population manager. Unlike the d2kSimGL and evStandaloneGL programs, the pmStandalone program does not generate any graphic images of the robot dynamics. The pmStandalone program is activated by typing "`pmStandalone pm.p eval.p`" at the prompt of a Unix console window.

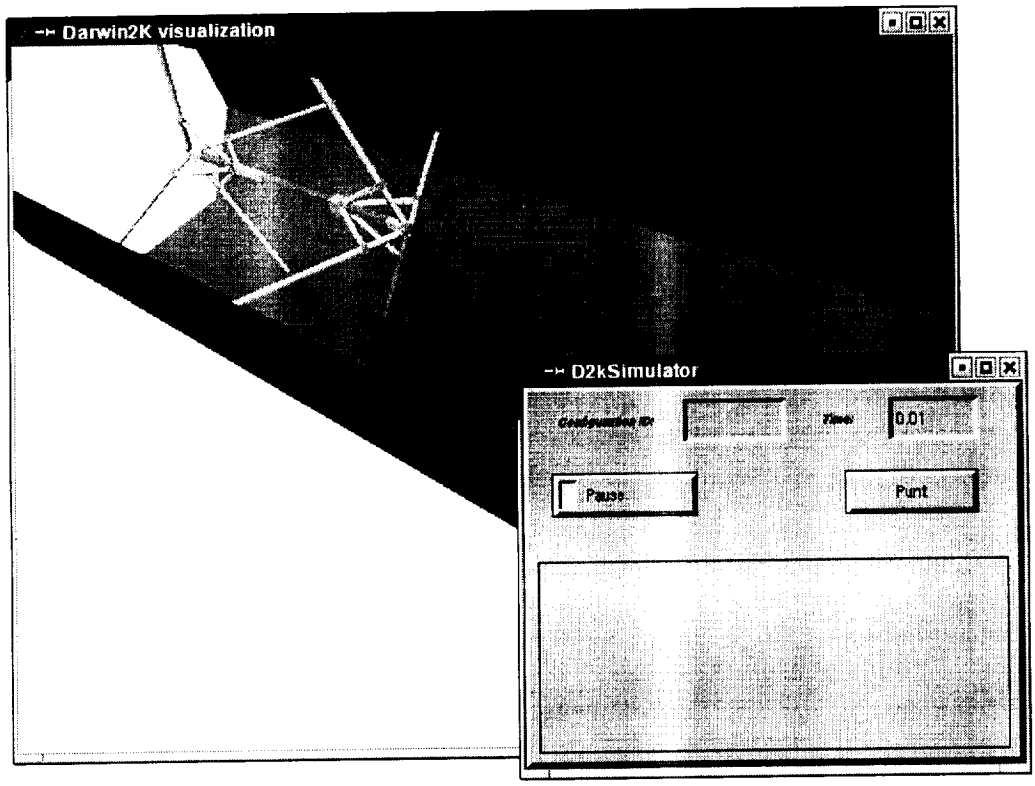

Figure 4: Darwin2K Simulation of UmbrellaBot Design

7.0 Results

The Darwin2K population manager was used to optimize a portion of the configuration of the UmbrellaBot. Many analytical iterations were necessary in order to properly configure Darwin2K to perform the optimization. The input files required for the Darwin2K population manager include the robot configuration file, the evaluation parameter file, the performance metric parameter file, and the terrain definition file. The task given to the robot was to move one unit of length down the long axis of the tube, which generally required one complete motion cycle of the robot. The robot configuration was measured in five categories of performance: 1.) completion of the task, 2.) the mass of robot, 3.) the time required to complete the task, 4.) the power consumed by the robot, and 5.) the deflection of the robot's links. The program manager was allowed to run through 10 generations with 10 population members each, for a total of 100 configurations. The population manager was allowed to vary (within limits) the length of the center prismatic joint and the length of the long "hollowTube" portion of the robot's legs. Three "optimal" robot configurations have been generated by the Darwin2K program manager. A listing of the population manager results is presented in Figure 5.

```
Logged population.
100
  totalCfgs      10
  optimalCfgs    3
  feasibleCfgs   10
  evaluations    0
  weights: [0] 0.000
  100 [0] 1.000000 taskCompletionMetric
best = 1.000
  100 [1] 0.000233 massMetric
best = 1.139
  100 [2] 0.000583 timeMetric
best = 0.583
  100 [3] 0.242532 powerMetric
best = 281.050
  100 [4] 0.001825 linkDeflectionMetric
best = 0.210
Generated 100 configurations, max = 100
Done.
```

Figure 5 – Output Log of the Darwin2K pmStandalone Program

The population manager calculated that, for the time performance metric, the best length of the center prismatic joint is 1.1875, the length of the

front leg hollowTube link is 1.4125, and the length of the rear leg hollow Tube is 1.405 (units are not provided since the robot's configuration was defined relative only to the tube). It is highly unlikely that this solution would have been reached using classical analysis methods. Provided with more resources, other parameters could be optimized, including the robot's controller gains or the number of legs for robot locomotion (or both). One limitation of Darwin2K involved the presentation and interpretation of results generated by the population manager. In its current, native mode, Darwin2K does not keep a running tab of the performance indeces of the robot configurations. This leads to many natural questions regarding the effectiveness of the population manager.

8.0 Summary

A shareware software toolkit developed for robot simulation and optimization, Darwin2K, has been used to investigate and refine the conceptual design of a robot. The design of a tube crawling robot is presented. A brief overview of Darwin2K is discussed, and the modifications made to Darwin2K in order to perform the analyses are documented. The evolutionary analysis tools provided by Darwin2K have been used to optimize portions of the robot's configuration for certain performance metrics.

The Darwin2K software toolkit has great potential for automating robotic design synthesis and optimization. However, the relative difficulty in using Darwin2K in its present form probably precludes its widespread adoption until several "ease of use" issues are addressed. However, the primary author of the software is presently working to address these issues.

The tube crawler robot whose design was simulated and ultimately refined using Darwin2K has been developed to the point that a more complete analysis is now feasible. Additional design studies will be required to determine appropriate sensors, power supplies, and useability constraints.

References

[1] Lipson, H. and Pollack, J., "Automated design and manufacture of robotic lifeforms," Nature, Vol 406, Macmillan Magazines, Ltd, August 31, 2000.

[2] Perkins, S. and Hayes, G., "Robot Shaping - Principles, Methods and Architectures," Workshop on Learning in Robots and Animals, AISB '96, University of Sussex, UK, April 1-2, 1996.

[3] Leger, C. "Darwin2K: An Evolutionary Approach to Automated Design for Robotics," Kluwer Academic Publishers, 2000.